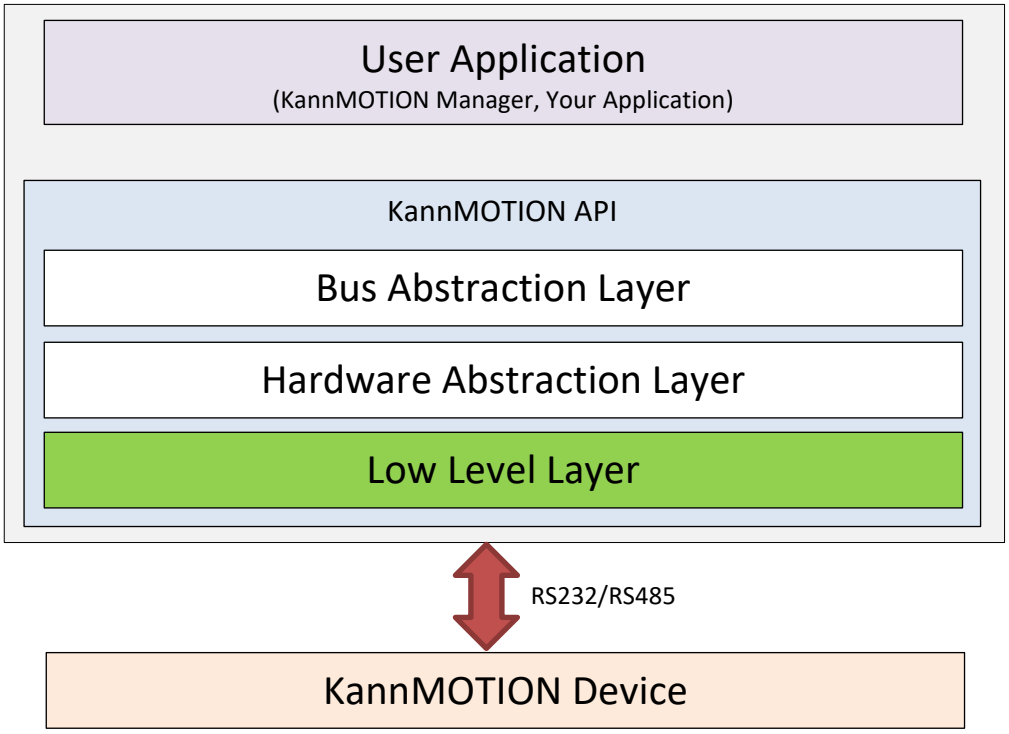
AN300045: Using Adlos Communication LowLevel API [LLL]

Introduction

The KannMOTION-API serves the easy connection of KannMOTION Devices with your application.
The API is split into 3 layers, the Low Level Layer (LLL), the Hardware Abstraction Layer (HAL) and the Bus Abstraction Layer (BAL).

This particular document introduces the user to the LLL-API. The LLL library allows you to work with some basic functions as read/write oprations, with protocol and checksum handling already integrated.

Common Infos



The LLL is available as a .NET Core library.

API

With this API you can get a simple Connection. With this Object, you can communicate with your KannMOTION device. Below a quick overview.

```
namespace Adlos.KMAPI.LLL
{
   public class Connection : Adlos.KM.Framework.Interfaces.Devices.IConnection, IDisposable
   {
      public const int BAUDRATE = 38400;
      public const int DEFAULT_SERIAL_TIMEOUT_FOR_MESSAGES_MS = 500;
      public const byte ECHO_LENGTH = 3;
      public const byte CHECKSUM_LENGTH = 1;
      public const byte TRIGGER_SIGN = 13;
      public const byte TRIGGER_INDEX = 2;

      public Connection(string port);

      public string PortName { get; }
      public int SerialTimeout { get; set; }

      public event EventHandler<KeyValuePair<string, List<byte>>> EventReceived;

      public void ActivateLogMode(string path = null);
      public void Close();
      public void Dispose();
      public void Open();
      public List<byte> Send(List<byte> command);
      public List<byte> Send(string command);
      public List<byte> Send(List<byte> command, int respBytes);
      public List<byte> Send(string command, int respBytes);
      public void SendWithoutResponse(List<byte> command);
      public void SendWithoutResponse(string command);
      public bool SubscribeEvent(string trigger, byte numOfBytes);
      public override string ToString();
      public bool UnsubscribeEvent(string trigger);
   }
}
```

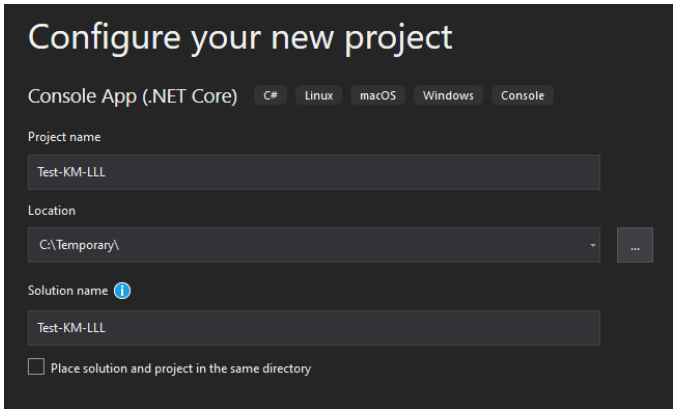
Usage

In this app note it is described, how to use the LLL DLL in a Visual Studio Project.
For this, we create a new Visual Studio Project (Console App .NET Core), include the LLL library and call some basic commands with API methods.
We will create a connection, read some device information and rotate and stop.

This example project is available as *Test-KM-LLL,* and can be opened with Visual Studio.
LLL version in example project is *0.9.4.14*.

Additional infos for protocol, API, etc. can be found under *Documentation & Links*.

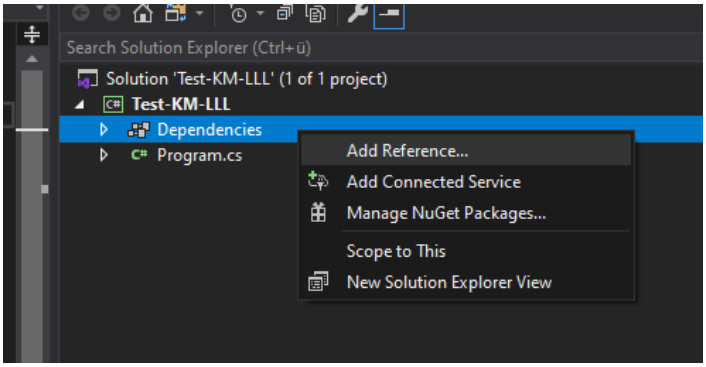
Create new Visual Studio Project



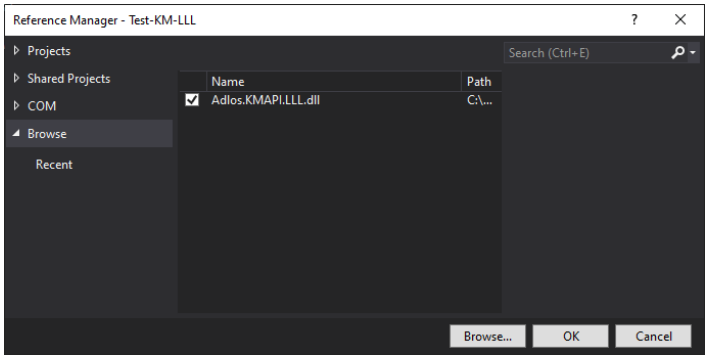
Then, click on **Create**.

Include LLL library

In your project map on the right side, right click on **Dependencies** and **Add Reference**.
In other versions of Visual Studio it can be called **Add Shared Project Reference**.



Next, go to **Browse** on the left side, click on **Browse** and select Adlos.KMAPI.LLL.dll and click **Add**.



Install needed NuGet

The LLL needs the *System.IO.Ports* NuGet. Again right click on **Dependencies** and click **Manage NuGet Packages...**
Search for System.IO.Ports and install it.

Examples

Here you can find some coded examples to get to work with the LLL API library.
You can find these examples in the *Test-KM-LLL* Visual Studio Project.

Code: Using directive

Allow the use of types in the namespace so that you do not have to qualify the use of the type in that namespace.

```
using Adlos.KMAPI.LLL;
```

Code: Get LLL Version

```
// Get version of LLL library
Version LllVersion = typeof(Connection).Assembly.GetName().Version;
```

Code: Connect to a KannMOTION device

First, you have to find out what COM-Port your device is on. Then make a new connection.

```
// Establish a connection with com port 25
string comPort = "COM25";
Connection KMConnection = new Connection(comPort);
```

Code: Read and format firmware article number and version

WIth command ID#0D the firmware articlenumber and version is read. To make it readable, the response hast o be formatted.

```
// Read fw article number and version with ID command
List<byte> idResponse = KMConnection.Send("ID#0D", 9);
string fwArtNr = new ASCIIEncoding().GetString(idResponse.ToArray()).Substring(0, 6);
Version fwVersion = new Version(idResponse[6] >> 4, idResponse[6] & 0x0F, idResponse[7]);
```

Code: Rotate with 100rpm for 5sec and stop

```
// Rotate with 100rpm for 5 sec then stop (1000 * 0.1rpm -> 0x03E8)
Console.WriteLine("Action:\t\tRotate with 100rpm");
KMConnection.Send("Dr#0D#E8#03#CK", 0);

Console.WriteLine("Action:\t\t...for 5sec...");
Task.Delay(5000).Wait();

Console.WriteLine("Action:\t\tStop rotating.");
KMConnection.Send("Dr#0D#00#00#CK", 0);
```

Code: Close connection

Clode the connection.

```
// Close connection
KMConnection.Close();
```

Or close the connection and free all used system resources.

```
// ...or close connection and free all used system resources
KMConnection.Dispose();
```

Code: Subscribe to event

It is possible, to send data from user sequence of a device and to subscribe to this.

```
// Subscribe to event with trigger "DF#0D" and 5 byte length (4 bate data, 1 CK)
KMConnection.SubscribeEvent("DF#0D", 5);
```

To get get the data, register a callback

```
// Register callback
KMConnection.EventReceived += Connection_EventReceived;
```

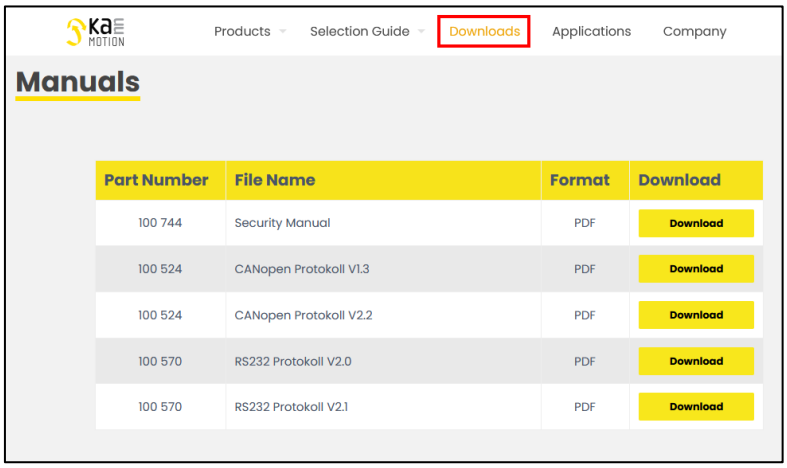
Within the callback, the event can be checked and read.
args.key ist he trigger, args.value the data (List<byte> of whole message).

```
private static void Connection_EventReceived(object sender, KeyValuePair<string, List<byte>> args)
{
    if (args.Key == "DF#0D")
    {
        byte[] data = { args.Value[3], args.Value[4], args.Value[5], args.Value[6] };
        Console.WriteLine("Received:\t{0}", BitConverter.ToUInt32(data));
    }
}
```

Commands (GoToPos, Rotate, etc.) can only be sent to the device, as long as there is no user sequence programed.

Documentation & Links

API, Protocols and additional information can be found on kannmotion.com under downloads tab.



Art. Nr.	*Description*
100570	RS232 Protocol
100524	CANopen Protocol
300054	Protocol Simplification
190073	LLL API (Low Level Layer)
190074	HAL API (Hardware Abstraction Layer)
190080	BAL API (Bus Abstraction Layer)
300045	LLL Application Note
300046	HAL Application Note
300047	BAL Application Note

Additional Files	
LLL Sample Code	https://kannmotion.li/download/ANs/300045/AN300045_SampleCode.zip

Contact information

Adlos AG
Föhrenweg 14
FL-9496 Balzers

Thomas Vogt
Thomas.Vogt@adlos.com
Tel: +423 263 63 63

Countries: CH, A, LI, SK, IT
www.adlos.com

KOCO MOTION GmbH
Niedereschacher Straße 54
D-78083 Dauchingen

Olaf Kämmerling
O.Kaemmerling@kocomotion.de
Tel: +49 7720/995858-0

Countries: DE, BE, NL, LU
www.kocomotion.de